Differential Equations, Fall 2016
Computer Project 2, Due Monday, October 3

The goal of this assignment is to implement the Euler Method and the Improved Euler Method, and to use these methods to numerically solve some differential equations. You should hand in the following things:

1. A printout of the program `EulerMethod.m` from problem 1.
2. A printout of the plots from problem 2.
3. A printout of the program `ImprovedEulerMethod.m`, and the plots from problem 3.
4. A printout of commands and output from problem 4.
5. A printout of the plot from problem 5.

Here is a warm-up exercise to teach you about functions with multiple return values. Do not hand it in. Create a function called `test2.m` in your directory (I recommend creating a separate directory for each computer project). Enter the following code into the function:

```
function [a b]=test2()
    a=[5 6];
    b=10;
end
```

The function takes in 0 inputs and returns 2 outputs, `a` and `b`. You can run it from the command line by typing in

```
> test2()
```

You can save its outputs to some variables by typing in

```
> [x y]=test2();
```

This will store the value of the first output into `x` and the value of the second ouput into `y`. If you type in

```
> z=test2();
```

only the first output will be stored into the variable `z`. You can create functions with an arbitrary number of outputs. For example, if the function began as

```
function [a b c]=test2()
```

then it would have 3 outputs.

1. (3 points) The goal of this problem is to write a program to implement the Euler Method to solve an initial value problem of the form

$$\begin{aligned} \frac{dy}{dx} &= f(x,y) \\ y(x_0) &= y_0. \end{aligned}$$

We will take $f(x,y) = y$ to begin with. First, we need to create a Matlab function for $f(x,y)$. To do this, create a function called `f.m` and enter the following code.

```
function val=f(x,y)
    val=y;
end
```

Test that it works by entering the command

```
> f(2,3)
```

(This should return 3.)

Now we will create the main program to implement the Euler Method. Create a function called `EulerMethod.m`. Start with the following code in this function:

```
function [xvals yvals]=EulerMethod(xmin,xmax,steps,y0)
% calculate the step size
h=(xmax-xmin)/steps;
% now set up a list of x values
% need the list to have steps+1 numbers in it
% because the initial x is counted as 1
xvals=linspace(xmin,xmax,steps+1);
x=xmin;
% initialize y and yvals
y=y0;
yvals=[y];
% now a for loop to build the yvals
for i=1:steps
    % x and y already have values stored in them
    % use these values to find slope
    slope=f(x,y);
    % now find the next y value
    y=...
    % and add it to the list of yvals
    yvals=[yvals y];
    % update x
    x=xvals(i+1);
end
end
```

The function takes in 4 inputs. The first input is the initial x value, the second input is the final x value, the third input is the number of steps to use, and the fourth input is the initial y value.

The function returns 2 outputs. The first output is a list of the x values, and the second output is a list of the corresponding y values. These values are the approximate numerical solution of the differential equation.

Lines that start with `%` are comment lines. They have no effect on the code and I put them in there to tell you a little bit about what the code is doing. The only thing you need to do to complete the program is to remove the ... and put in the correct formula for `y` in the for loop.

Print out the program `EulerMethod.m`.

2. (3 points) In this problem we will run the function `EulerMethod` you created in the previous problem and plot the output. Note that the differential equation we are considering for now is $y' = y$, the solutions of which are $y = Ce^x$. To consider a different equation we would need to change the function `f.m`.

   We will take the initial condition $y(0) = 1$, and use the program to find an approximate solution for $0 \leq x \leq 2$. Run the command

   ```
   > [x y]=EulerMethod(0,2,5,1);
   ```

   This will give a crude approximation using 5 steps. It should look roughly like the exponential function. We can graph it by running the command

   ```
   > plot(x,y)
   ```

   Now run the command

   ```
   > [x y]=EulerMethod(0,2,1000,1);
   ```

   and graph it using the commands

   ```
   > hold on
   > plot(x,y)
   ```

   The `hold on` command tells Matlab to draw all new plots into the current figure. The `hold on` command will remain active until you close the figure or turn it off with the command `hold off`. (At least this is the way Matlab seemed to work on my computer. Octave works a little differently.) Drawing both plots in the same figure will allow you to see the difference between the two solutions. Matlab will generally use a new color when making a new plot in the same figure. You can can control the color by passing it into the plot function; for example, `plot(x,y,'r')` will plot in red. You can enter the command `help plot` to see a list of available colors. Print out the figure containing both solutions (the printout does not need to be in color).

3. (3 points) Now create a file called `ImprovedEulerMethod.m` to implement the Improved Euler Method. You can start by reusing the outline of the previous code. Here it is to get you started:

```
function [xvals yvals]=ImprovedEulerMethod(xmin,xmax,steps,y0)
% calculate the step size
h=(xmax-xmin)/steps;
% now set up a list of x values
% need the list to have steps+1 numbers in it
% because the initial x is counted as 1
xvals=linspace(xmin,xmax,steps+1);
x=xmin;
% initialize y and yvals
y=y0;
yvals=[y];
% now a for loop to build the yvals
for i=1:steps
    % x and y already have values stored in them
    % use these values to find slope
    slope1=f(x,y);

    ...

    y=...;
    yvals=[yvals y];
    x=xvals(i+1);
end
end
```

Now we will test that it works by plotting the output as in the previous problem. First, if the figure from the previous exercise is still open, close it. Then run the commands

```
> hold on
> [x y]=ImprovedEulerMethod(0,2,5,1);
> plot(x y)
> [x y]=ImprovedEulerMethod(0,2,1000,1);
> plot(x y)
```

This should draw both plots in the same figure. Print out the figure containing both plots. Also, print out the program `ImprovedEulerMethod.m`.

You should notice that the solutions are much closer to each other than the ones in the previous exercise were. This suggests that they are closer to the actual exact solution. If you want to, you can plot the solutions from the previous exercise in the same figure to see the difference between the two methods. You do not need to turn it in.

4. (3 points) Now we will test how accurate the programs are by using them to calculate the number e. You can get Matlab to display the exact value of e to several decimal places by entering the commands

```
> format long
> exp(1)
```

Now run the following commands to get several approximations using your program

```
> [x y]=EulerMethod(0,1,10,1); y(11)
> [x y]=EulerMethod(0,1,1000,1); y(1001)
> [x y]=EulerMethod(0,1,100000,1); y(100001)
> [x y]=ImprovedEulerMethod(0,1,10,1); y(11)
> [x y]=ImprovedEulerMethod(0,1,1000,1); y(1001)
> [x y]=ImprovedEulerMethod(0,1,100000,1); y(100001)
```

Print out your commands along with the output.

5. (3 points) Now we will consider the equation

$$\frac{dy}{dx} = y(1-y) + .1\sin(x).$$

Use the Improved Euler Method with 10000 steps to solve this equation on the interval $0 \leq x \leq 15$ For the initial condition, first take $y(0) = .25$, and then $y(0) = .5$, and then $y(0) = .75$, and so on up to $y(0) = 2$. So you will have solved the equation 8 times with 8 different initial conditions. Plot all the solutions in the same figure and print it out.

Here is something else to think about (you do not need to hand in an answer): What happens when you take an initial condition of $y(0) = -1$? Can you explain why?